(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

SIMULATION OF ADJACENT CHANNEL INTERFERENCE IN A UHF SATELLITE SYSTEM

by

Juan Carlos Minuto

September, 1993

Thesis Advisor: Dr. Paul H. Moose

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*

2. REPORT DATE
September 23, 1993

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
SIMULATION OF ADJACENT CHANNEL INTERFERENCE IN A UHF SATELLITE SYSTEM

5. FUNDING NUMBERS

6. AUTHOR(S)
Minuto, Juan Carlos

7. PERFORMING ORGANIZATION NAMES (S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Apprcved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In this thesis, the adjacent channel interference in a ultra high frequency (UHF) satellite channel is evaluated by simulation and differential binary phase-shift keying (DBPSK) is compared with continous phase frequency-shift keying (CPFSK). First, a measure of the interfering power is obtained and a method to compute carrier-to-interference ratios in a non-linear channel is developed. Next, a DBPSK receiver is simulated when two interfering channels separated in frequency are present, and bit errors are detected and counted. Then, coherent reception of minimum-shift keying (MSK) and CPFSK with modulation index h=0.4 are simulated in the same conditions as DBPSK. Finally, noncoherent MSK is analyzed in the same way and a comparative behavior is obtained. It is found that the best performance in the presence of adjacent channel interference is given by coherent reception of MSK.

14. SUBJECT TERMS
DBPSK, CPFSK, MSK

16. PRICE CODE

15. NUMBER OF PAGES
71

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18, 298-102

Approved for public release; distribution is unlimited.

SIMULATION OF ADJACENT CHANNEL INTERFERENCE IN A UHF SATELLITE SYSTEM

by

Juan Carlos Minuto
Lieutenant, Argentine Navy

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING
(Electronic Warfare)

from the

NAVAL POSTGRADUATE SCHOOL
September, 1993

Author: ______________________
Juan Carlos Minuto

Approved By: ______________________
Dr. Paul H. Moose, Thesis Advisor

____
Dr. R. Clark Robertson, Second Reader

____
Jeffrey B. Knorr, Chairman
Electronic Warfare Academic Group

ABSTRACT

In this thesis, the adjacent channel interference in a ultra high frequency (UHF) satellite channel is evaluated by simulation and differential binary phase-shift keying (DBPSK) is compared with continous phase frequency-shift keying (CPFSK). First, a measure of the interfering power is obtained and a method to compute carrier-to-interference ratios in a non-linear channel is developed. Next, a DBPSK receiver is simulated when two interfering channels separated in frequency are present, and bit errors are detected and counted. Then, coherent reception of minimum-shift keying (MSK) and CPFSK with modulation index h=0.4 are simulated in the same conditions as DBPSK. Finally, noncoherent MSK is analyzed in the same way and a comparative behavior is obtained. It is found that the best performance in the presence of adjacent channel interference is given by coherent reception of MSK.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

This thesis is the work of many people other than myself. I especially appreciate Dr. Paul H. Moose for his support and help. Without him and his insightful ideas this work would not have been possible.

I also would like to thank Dr. R. Clark Robertson and Dr. Ralph Hippenstiel for their positive attitude towards my investigation. They made me feel free to ask questions that helped me in the process of developing this thesis.

Finally, I would like to make a special mention to all the professors who cooperated in my education at the Naval Postgraduate School. They stimulated my interest in the communications area, providing the necessary background to face this task with the adequate knowledge.

I. INTRODUCTION

A. DISCUSSION

The main goal of the ultra high frequency (UHF) satellite system is to provide reliable data transmission between multiple mobile users. In a digital satellite system, performance is measured in terms of the average probability of bit error. Given a sufficiently large bit-energy-to-single-sided-noise-power-spectral-density ratio (Eb/No), which is directly proportional to the carrier-to-noise ratio (C/N), it is generally assumed that the probability of bit error (Pb) [Ref. 1], can be made arbitrarily small. The UHF satellite is a frequency-division multiple access (FDMA) system. Consequently, when a second user accesses an adjacent channel, some spillover, called adjacent channel interference, will occur, and this will degrade the performance of the system, even for large C/N, since the effect of adjacent channel interference is to reduce C/N.

B. INTERFERING SOURCES -- GENERAL CONSIDERATIONS

If the interference source is assumed to be a statistically independent wide-sense stationary random process of zero mean, the overall carrier-to-noise-plus-interference ratio can be expressed by [Ref. 2],

$$\frac{C}{K} = \left[\left(\frac{C}{N}\right)^{-1} + \left(\frac{C}{I}\right)^{-1}\right]^{-1}, \tag{1}$$

where C/N is the carrier-to-noise ratio of the overall link, and C/I is the carrier-to-interference ratio of the overall link. When the interferences are non-Gaussian but numerous and none of them has a dominant effect, their joint probability density function approaches the Gaussian probability density function as stated by the central limit theorem. The effect of interference in this case can therefore be assumed to be equivalent to the effect produced by a single additive white gaussian noise (AWGN) process with the same carrier-to-interference ratio. The treatment of non-Gaussian interferences as equivalent AWGN generally results in a higher predicted probability of bit error than occurs in practice, probably because the sources are not Guassian and because they are not sufficiently numerous for the central limit theorem to apply.

The consideration of interference in satellite systems is of utmost importance. The interference could come from such different sources as adjacent satellite systems, terrestrial interference, cross-polarization interference, adjacent channel interference, and intermodulation interference.

Adjacent satellite system interference is generated by an earth station different than the one under consideration, and is caused by the power received through the antenna sidelobes which interferes with the main transmission. This effect can only be overcome by designing an antenna with smaller sidelobes.

Terrestrial interference is caused by terrestrial networks working in frequency bands where satellite systems have channels allocated. In the case of the UHF satellite channel, the interference could come from, for example, terrestrial mobile systems or harbor navigation systems. It is known that these kind of networks have a limited range, but in certain conditions, such as surface ducts, the transmission might reach unexpected distances and therefore interfere with a satellite earth station that is located outside the area of influence of the interfering sources.

Cross-polarization interference is produced in satellite systems in which orthogonal linear polarizations are employed to allow frequency reuse. The depolarization effect caused by rain and the finite cross-polarization discrimination of the earth station allow the channels to interfere with one another in spite of the orthogonal polarization condition in the transmission of the communication message.

Intermodulation interference is caused by the intermodulation products generated within a satellite transponder as a result of the non-linear amplification of multiple carriers by the traveling wave tube amplifier (TWTA). By operating the high power amplifiers at a certain output backoff, one can reduce their non-linear effect and reduce the intermodulation interference.

1. Adjacent Channel Interference

Another source of interference in a FDMA satellite link is the adjacent channel interference. For example, the power spectral density of binary phase-shift keying (BPSK) is represented in Figure 1, and it can be seen that most of the energy is concentrated in the main lobe which occupies a bandwidth $B = 2/Tb$ where Tb is the bit duration. However, the sidelobes of the spectrum contain some energy and if not properly filtered out, they can interfere with adjacent channels provided the separation between them is not high enough. This situation is depicted in Figure 2.

Obviously, a modulation scheme with smaller sidelobes will have a better performance, as far as adjacent channel interference is concerned, than one with higher sidelobes. A modulation scheme with a very compact mainlobe and low sidelobes is minimum-shift keying (MSK), which belongs to the family of continuous phase modulation schemes with a modulation index h=0.5. The basis of this work will be a comparative analysis of the adjacent channel interference between differential binary phase-shift keying (DBPSK) and continuous phase frequency-shift keying (CPFSK), a form of continuous phase modulation.

2. Jamming Considerations

The interference coming from a jammer can be considered in the same way as interference from unintentional sources. That is, since

$$\frac{C}{\aleph} = \frac{C}{N+J} \tag{2}$$

where J=jamming energy. Then

$$\frac{C}{\aleph} = [(\tfrac{C}{N})^{-1} + (\tfrac{C}{J})^{-1}]^{-1}, \tag{3}$$

where C/J is the carrier-to-jamming ratio.

Including the interference, we get

$$\frac{C}{N} = [(\tfrac{C}{N})^{-1} + (\tfrac{C}{I})^{-1} + (\tfrac{C}{J})^{-1}]^{-1}. \tag{4}$$

The term $(C/J)^{-1}$ is called the jamming margin and is the amount of jamming the system can tolerate for a certain probability of bit error. Since for a given Pb and modulation type, a unique value of $\frac{C}{N}$ is required, it also determines the C/J and C/I the system can accept without significantly degrading its performance.

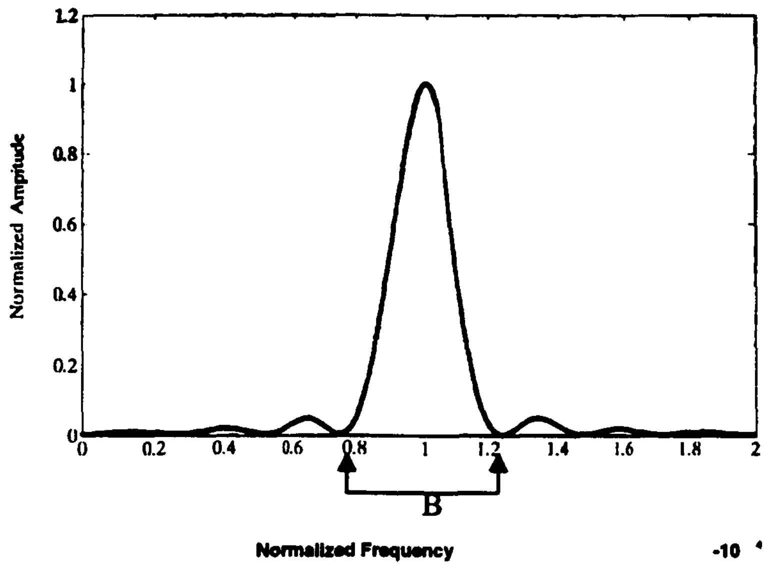

Figure 1. Power Spectral Density of BPSK

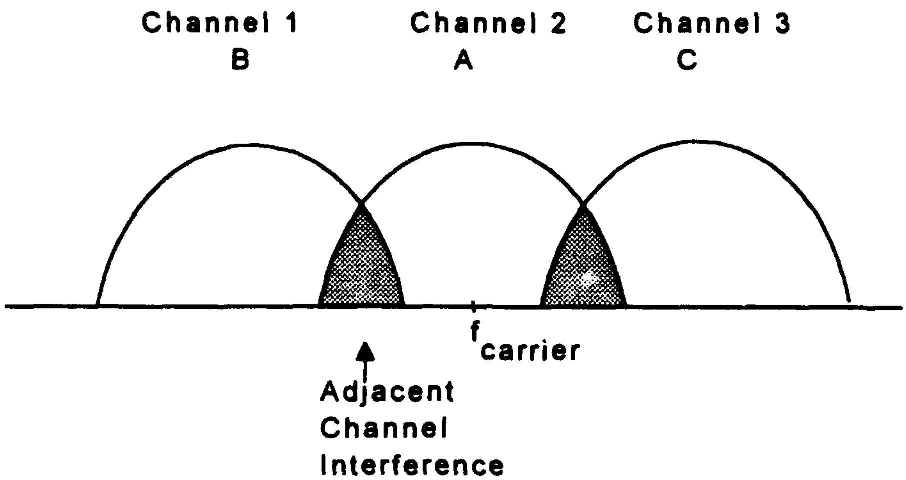

Figure 2. Adjacent Channel Interference.

C. OBJECTIVE

At present, the UHF satellite described in the Hughes Aircraft Company Space and Communication Group proposal [Ref. 3] cannot successfully be used at bit rates of 4800 bps or higher. The objective of this thesis is to demonstrate that this limitation is due to adjacent channel interference and can be solved by using a modulation scheme other than DBPSK such as MSK.

II. ANALYSIS OF INTERFERING POWER

A. UHF SATELLITE MODEL

The basic model of the satellite channel that was used to run all the simulations contained in this thesis is shown in Figure 3. The key modules were adopted from [Ref. 3]. They consist of a prelimiter filter, a hard limiter filter, and a postlimiter filter.

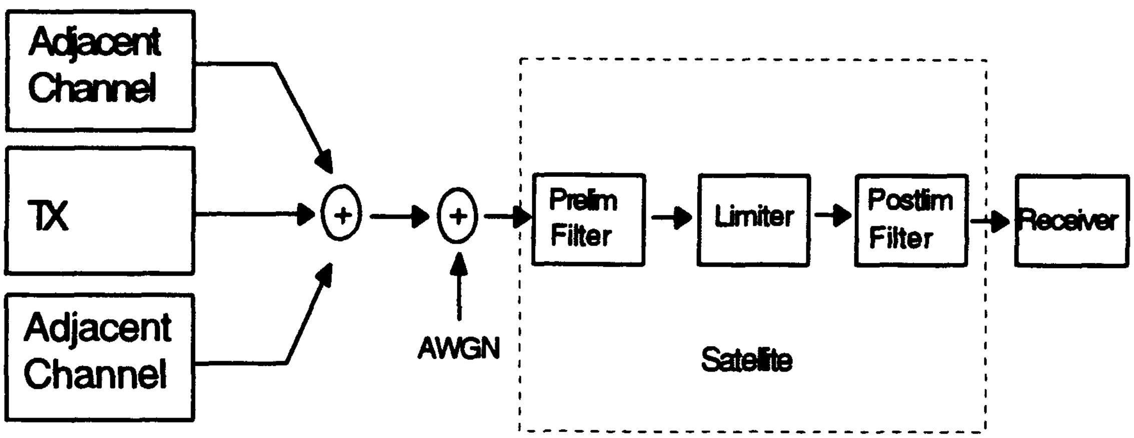

Figure 3. Basic Satellite Model with Adjacent Channel Interference

1. Prelimiter Filter

The prelimiter filter was implemented as Chebyshev Filter with 6 poles and 0.01 dB passband ripple. All the simulations were implemented using MATLAB. MATLAB's filter function accepts a normalized cutoff frequency value between 0 and 1; 1 corresponds to half the sampling rate. For the baseband model of the satellite channel a sampling frequency (f_s) equal to 384 kHz was chosen. This high sampling frequency is required since $f_s > 2\, f_{max}$ is required to avoid aliasing. In this case f_{max} = 100 kHz is the upper frequency of the upper adjacent channel in the baseband simulation.

For an analog cutoff frequency of f_c, the digital cutoff frequency is

$$f_{cutoff} = \frac{2f_c}{f_s}\,. \tag{5}$$

The frequency response, both magnitude and phase, and the unit impulse response corresponding to this filter with an analog cutoff frequency f_c = 12.57 kHz are plotted in Figures 4, 5, and 6, respectively.

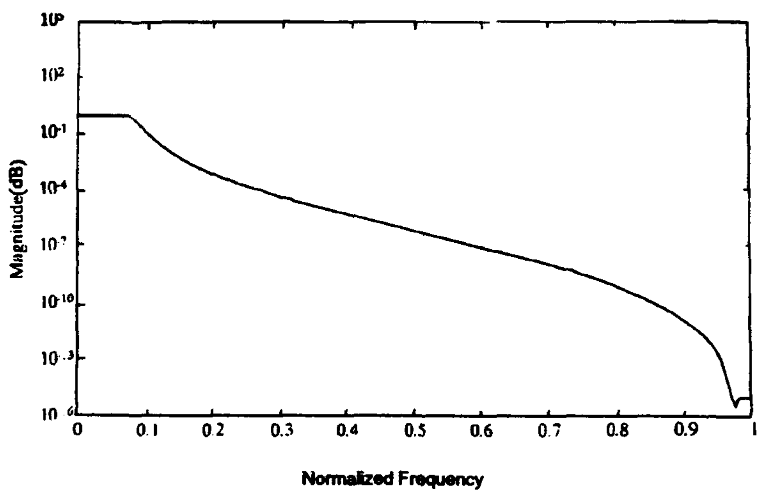

Figure 4. Frequency Response of the Prelimiter Filter.

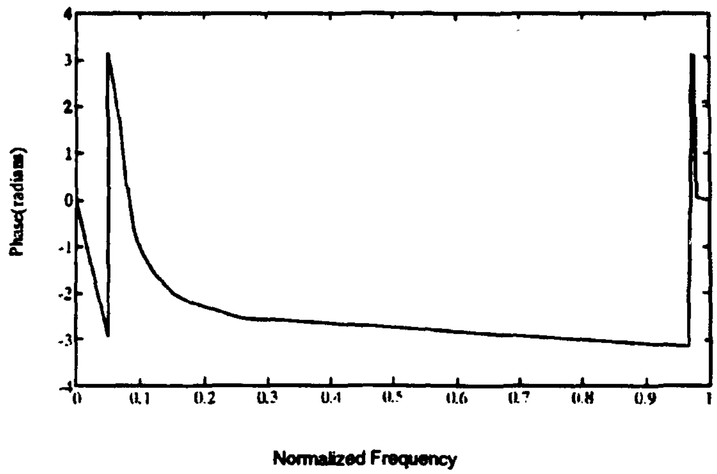

Figure 5. Phase Plot of Prelimiter.

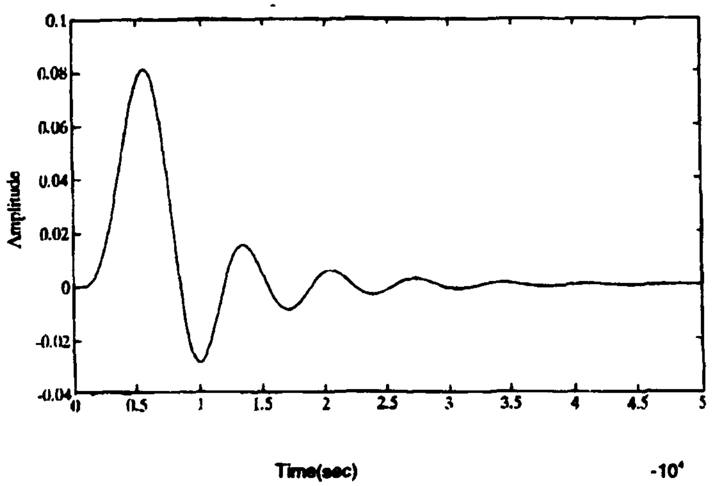

Figure 6. Unit Impulse Response of the Prelimiter Filter.

2. Hard Limiter

The hardlimiter is used to provide constant output power for input signal power varying from the noise threshold to maximum signal input. This was simulated by dividing each sample by its magnitude such that each complex sample is on the unit circle.

3. Postlimiter Filter

This filter was implemented as a Chebyshev filter, with 4 poles and 0.025 dB passband ripple. Based on the same considerations as before, the digital cutoff frequency for this filter is $f_{cutoff} = 0.0394$, since the analog cutoff frequency f_c is 7.56 kHz. The frequency response, both magnitude and phase, and the unit impulse response for this filter are plotted in Figures 7, 8, and 9, respectively.

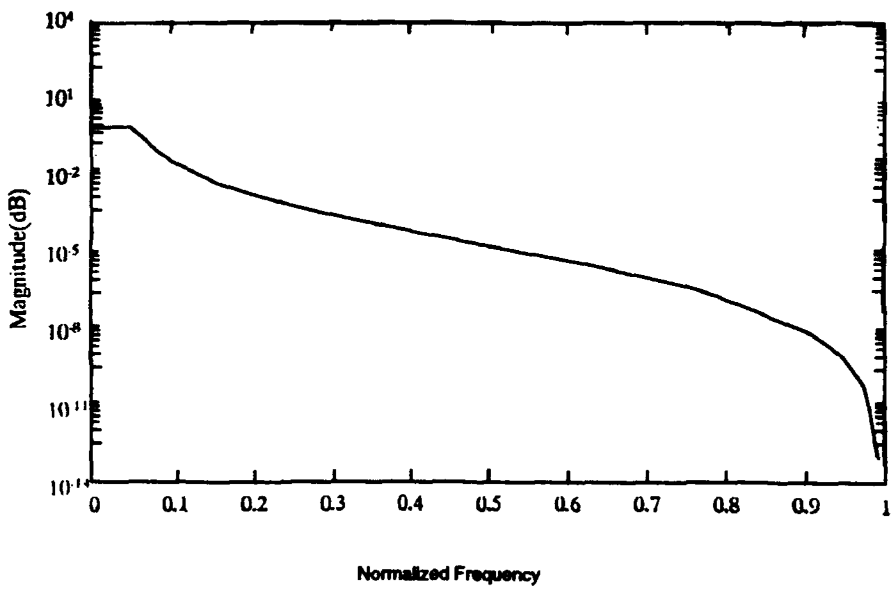

Figure 7. Frequency Response of the Postlimiter Filter.

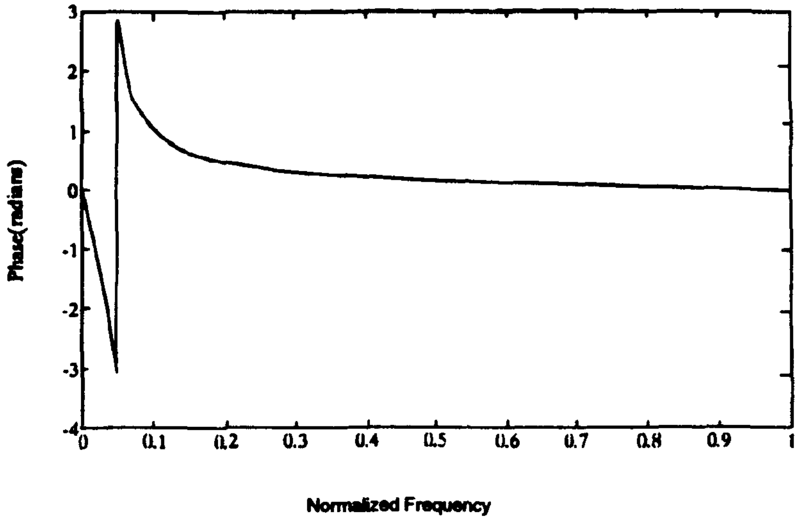

Figure 8. Phase Plot of the Postlimiter Filter.

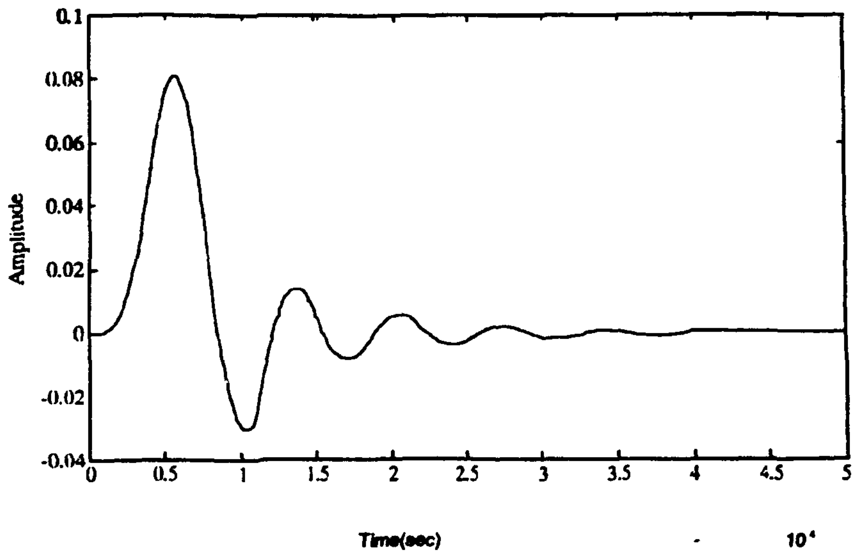

Figure 9. Unit Impulse Response of the Postlimiter Filter.

B. LINEAR ESTIMATION OF INTERFERING POWER

Having described the basic components of the UHF satellite channel we are now in a position to analyze the interference from adjacent channels. Initially, only interference due to the upper channel is considered. Consequently, consider the designated channel to be a baseband channel with $f_{carrier} = 0$ (see Figure 2). The results can be easily extended to more than one channel.

The separation in frequency between channels plays an important role. Not all the channels of the UHF satellite are equally spaced in frequency. The worst case, a frequency separation equal to 100 kHz, was used in the simulation. The first experiments used DBPSK as the modulation scheme. A block diagram for the experiment is shown in Figure 10.

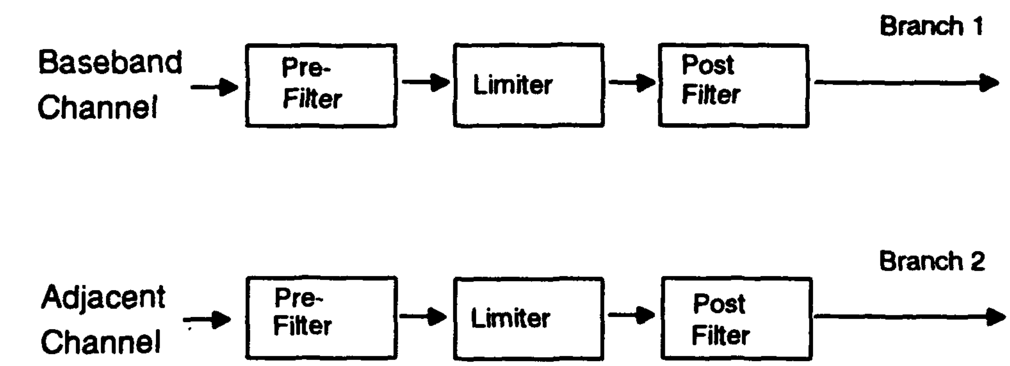

Figure 10. Block Diagram of Simulation.

From Branch 1, the power in the baseband from the on-channel signal was computed. Similarly from Branch 2, the power in the baseband coming from the adjacent channel was obtained. The results for this experiment are presented in Table 1. This experiment gives a first indication of the interfering effect, but it is not useful to provide an accurate value of the C/I ratio. No consideration was given to the correlation of the processes introduced by the limiter since both channels were analyzed separately.

TABLE 1. NORMALIZED BASEBAND AND ADJACENT POWER FOR DBPSK

	Bit Rate			
	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT POWER	0.11	0.19	0.33	0.43
BASEBAND POWER	0.97	0.94	0.89	0.77
BP/AP (dB)	9.37	6.96	4.31	2.56

A brief look at Table 1 shows that the figures obtained are as expected. For a higher bit rate the power spectral density of DBPSK is wider [Ref. 2]; more power from the adjacent channel and less of the baseband power is in the baseband channel bandwidth.

For the second experiment, continuous phase modulation was selected as a possible scheme for improvement with regard to adjacent channel interference. The same simulation was run, and the results for MSK (CPFSK with a modulation index h =1/2) are shown in Table 2.

TABLE 2. NORMALIZED BASEBAND AND ADJACENT POWER FOR MSK

	Bit Rate			
	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT POWER	0.09	0.15	0.27	0.36
BASEBAND POWER	0.99	0.99	0.99	0.97
BP/AP (dB)	10.67	8.19	5.67	4.27

Two other attempts were made to find out if a different modulation index h could improve performance. CPFSK, with indexes ranging from 0.1 to 1, was analyzed and the results are plotted in Figures 11 and 12. Similar performance is expected for MSK and CPFSK with $h = 0.4$. However, a small improvement can be detected at 19200 bps for $h=0.4$. Therefore, the simulation was run for CPFSK with $h = 0.4$ and the results can be seen in Table 3.

TABLE 3. NORMALIZED BASEBAND AND ADJACENT POWER FOR CPFSK WITH h=0.4

	Bit Rate			
	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT POWER	0.09	0.15	0.27	0.36
BASEBAND POWER	0.99	0.99	0.99	0.98
BP/AP (dB)	10.56	8.19	5.58	4.35

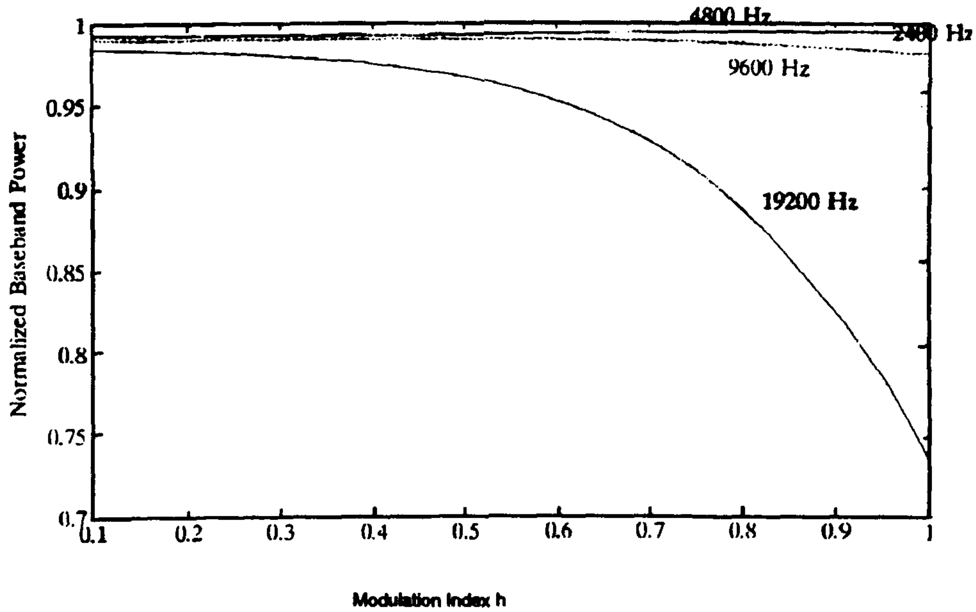

Figure 11. Baseband Power.

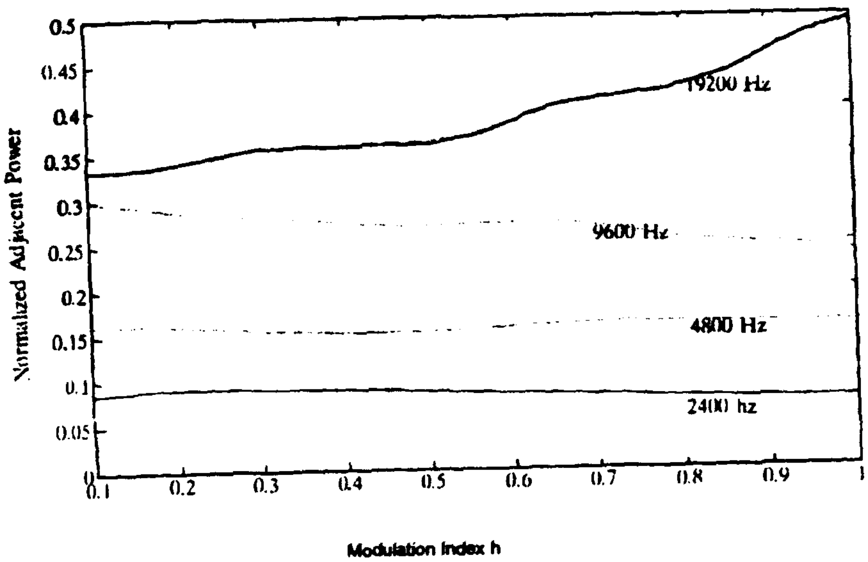

Figure 12. Adjacent Power.

The final experiment used Gaussian MSK as a modulation scheme. This particular type of modulation is fully described by Murota and Hirade [Ref. 4]. It is stated to have a better performance than MSK in certain aspects, such as ISI degradation. The simulation was therefore run for this particular case, and the results are shown in Table 4.

TABLE 4. NORMALIZED BASEBAND AND ADJACENT POWER FOR GAUSSIAN MSK

	Bit Rate			
	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT POWER	0.07	0.17	0.31	0.55
BASEBAND POWER	0.99	0.99	0.99	0.97
BP/AP (dB)	11.33	7.73	5.02	2.47

Comparing all the results obtained so far, it can be concluded that MSK and CPFSK with h = 0.4 are candidates to outperform DBPSK in the case of adjacent channel interference. Therefore, a more detailed study is necessary to obtain a more accurate estimate of the actual carrier-to-interference ratio. An approach to deal with this situation is developed in the next section.

C. CARRIER-TO-INTERFERENCE RATIO FOR A NON-LINEAR CHANNEL

Because of the presence of the hard limiter in the satellite, the system is not linear, and therefore a more accurate technique to estimate the carrier-to-interference ratio is necessary. The method chosen consists of estimating the on-channel signal and removing it from the on-channel plus interference signals in order to estimate the interference. The block diagram in Figure 13 illustrates this technique.

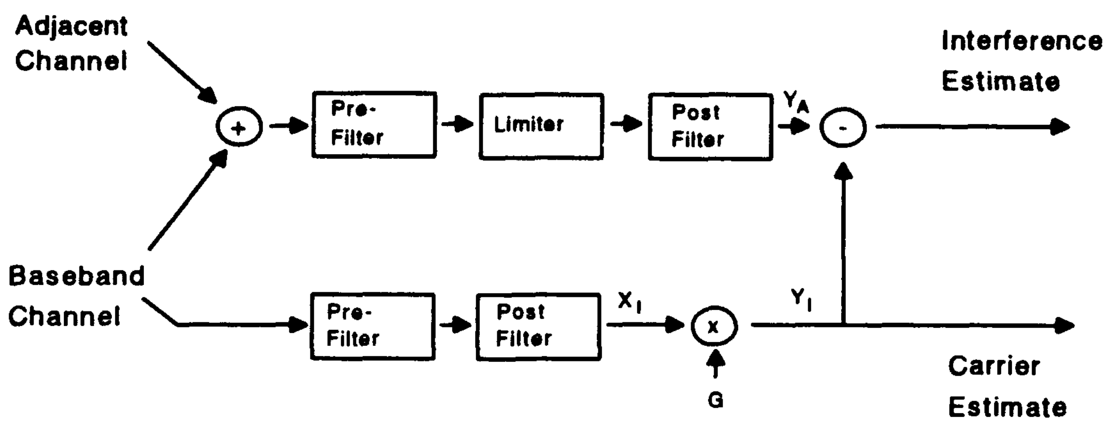

Figure 13. Block Diagram of Estimation Method.

Since the adjacent channel signal is being generated independently of the on-channel signal, the input processes are uncorrelated with one another. The best estimate, Y_I, of the on-channel signal in Y_A occurs when Y_I is orthogonal to the error, Y_A-Y_I. This is when

$$E[(Y_A - Y_I) \times Y_I] = 0 \quad , \tag{6}$$

which leads to

$$E[(Y_A - X_I \times G) \times X_I G] = 0 \quad , \tag{7}$$

and

$$G = \frac{E(Y_A \times X_I)}{E(X_I^2)} \quad . \tag{8}$$

The carrier-to-interference power ratio can then be expressed as

$$\frac{C}{I} = \frac{E[|X_I \times G|^2]}{E[|Y_A - X_I \times G|^2]}. \qquad (9)$$

This technique was used to estimate the carrier-to-interference ratio for DBPSK, MSK, and CPFSK with h = 0.4. The results are presented in Table 5.

TABLE 5. CARRIER TO INTERFERENCE RATIOS (IN dB) FOR DBPSK, MSK, AND CPFSK

	Bit Rate			
Mod. Scheme	2400 bps	4800 bps	9600 bps	19200 bps
DBPSK	25.5	22.66	18.14	16.24
MSK	30.48	27.26	24.48	19.89
CPFSK (h=0.4)	30.58	27.43	24.86	21.03

In the same way, and based on the independence assumption among channels, the carrier-to-interference ratio for two adjacent channels can be calculated. Table 6 shows the results for DBPSK and CPFSK (h = 0.4).

TABLE 6. CARRIER TO INTERFERENCE RATIOS (IN dB) FOR DBPSK AND CPFSK -- TWO ADJACENT CHANNELS

	Bit Rate			
Mod. Scheme	2400 bps	4,800 bps	9600 bps	19200 bps
DBPSK	22.23	19.09	14.89	12.46
CPFSK (h = 0.4)	27.58	24.43	21.78	17.98

The results in Table 6 were obtained by locating a lower interference channel 100 kHz from the baseband channel. The simulation was then run with both upper and lower interfering channels.

These results show that either MSK or CPFSK (h = 0.4) have C/I significantly higher than DBPSK at all data rates. The procedure can be continued by adding additional channels spaced in frequency by 100 kHz from the on-channel signal. However, it is assumed that the total adjacent channel interfering power is dominated by the first adjacent channels.

III. ANALYSIS OF THE SATELLITE CHANNEL FOR DIFFERENT MODULATION TECHNIQUES

A. DBPSK ANALYSIS AND SIMULATION RESULTS

For this modulation scheme, a model similar to the one used by Khanaman [Ref. 5] was simulated. First, a lower and upper interfering channel separated 100 kHz in frequency from the baseband channel were simulated. Since the computed carrier-to-interference ratio (C/I) for this case is very high (see Table 5), no errors were expected to be found due to the adjacent channels. The limitations imposed by the computer simulation run time (no more than 1000 bits were simulated) do not allow the channel to be analyzed in the region where the probability of bit error is expected to be as low as 10^{-6}. Therefore, it was decided to reduce the frequency separation so as to cause some errors to appear in order to have a measure to compare DBPSK and CPFSK.

Obviously, the count of the number of errors in any Monte Carlo simulation does not represent accurately the probability of bit error, because only a finite number of trials are possible. However, the number of errors can provide a good idea of comparative behaviour between two different modulations when the same parameters are used for the channels.

Consequently, a second simultation was run placing two adjacent channels at +/- 15 kHz and a third simulation was run locating the interfering sources at +/- 12.5 kHz. To have even more data to analyze, the channel was tested for three different Eb/No conditions: 14 dB, 12 dB, and 10 dB (in the last one +/- 25 kHz was used instead of +/- 15 kHz).

A block diagram of the channel is presented in Figure 14, and the simulation results are presented in Tables 7, 8, and 9. The different codes that were used to simulate DBPSK can be found in Appendix A.

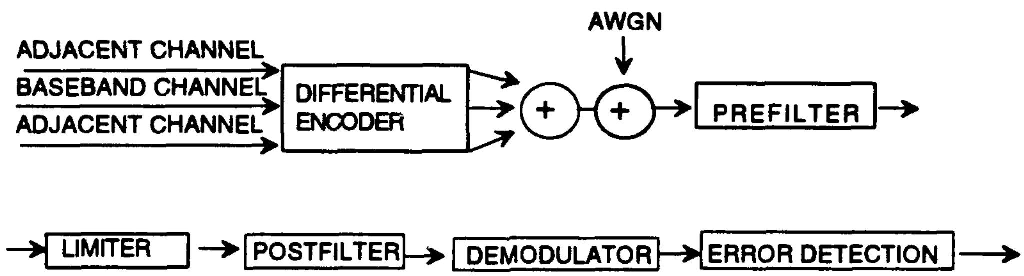

Figure 14. Block Diagram of DBPSK Satellite Channel Simulation.

TABLE 7. NUMBER OF ERRORS FOR THE SATELLITE CHANNEL WITH 14 dB Eb/No.

	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT CHANNELS (100 kHz)	0	0	0	0
ADJACENT CHANNELS (15 kHz)	0	0	0	90
ADJACENT CHANNELS (12.5 kHz)	0	0	1	147

TABLE 8. NUMBER OF ERRORS FOR THE SATELLITE CHANNEL WITH 12 dB Eb/No.

	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT CHANNELS (100 kHz)	0	0	0	2
ADJACENT CHANNELS (15 kHz)	0	0	2	100
ADJACENT CHANNELS (12.5 kHz)	0	1	4	148

TABLE 9. NUMBER OF ERRORS FOR THE SATELLITE CHANNEL WITH 10 dB Eb/No.

	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT CHANNELS (100 kHz)	0	0	0	3
ADJACENT CHANNELS (25 kHz)	0	0	0	12
ADJACENT CHANNELS (12.5 kHz)	1	1	8	152

B. MSK ANALYSIS AND SIMULATION RESULTS

1. Coherent Reception

In coherent MSK, it is assumed that the initial phase of the transmitted signal is perfectly known at the receiver. Two basic coherent receivers were modeled for this study. The first is explained by Haykin [Ref. 6]. Essentially, it consists of a correlator receiver with two branches where the decision is made by alternatively evaluating the signal after integrating it over a period equal to twice the bit duration (2 Tb) with one bit offset. A simplified block diagram can be seen in Figure 15. The decision logic is shown in Figure 16.

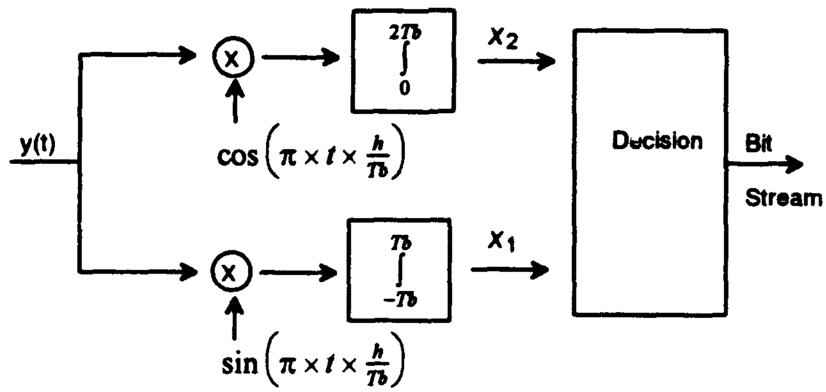

Figure 15. Coherent Demodulator of MSK.

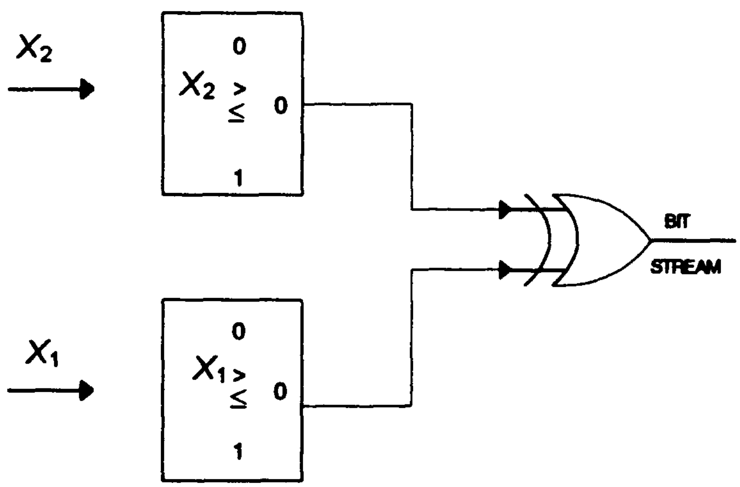

Figure 16. Decision Logic for MSK Receiver.

The same procedure was followed with DBPSK. The simulation was run placing the adjacent channels at +/- 100, +/- 15, and +/- 12.5 kHz. (As before, when a 10 dB Eb/No was used, the frequency spacing was +/- 25 kHz instead of +/- 15 kHz.) The results can be seen in Tables 10, 11, and 12. It follows that there is an improvement in the system if MSK is used because no errors were found until the channels were unacceptably close, and even in this situation the number of errors computed was considerably lower than in the case of DBPSK.

TABLE 10. NUMBER OF ERRORS FOR THE SATELLITE CHANNEL WITH 14 dB Eb/No.

	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT CHANNELS (100 kHz)	0	0	0	0
ADJACENT CHANNELS (15 kHz)	0	0	0	4
ADJACENT CHANNELS (12.5 kHz)	0	0	0	71

TABLE 11. NUMBER OF ERRORS FOR THE SATELLITE CHANNEL WITH 12 dB Eb/No.

	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT CHANNELS (100 kHz)	0	0	0	0
ADJACENT CHANNELS (15 kHz)	0	0	0	6
ADJACENT CHANNELS (12.5 kHz)	0	0	0	71

TABLE 12. NUMBER OF ERRORS FOR THE SATELLITE CHANNEL WITH 10 dB Eb/No.

	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT CHANNELS (100 kHz)	0	0	0	0
ADJACENT CHANNELS (25 kHz)	0	0	0	0
ADJACENT CHANNELS (12.5 kHz)	0	0	0	83

In the second receiver, the Viterbi algorithm is used to decode the MSK signal. As explained by Proakis [Ref. 7], the states for the phase of MSK can be +/- $\pi/2$, 0, and π. The number of states can be reduce if the signal is premultiplied by $e^{j\pi hn}$. Note that an MSK signal leaving from a phase of zero will increase the phase by $\pi/2$ if the input is a logical "one." The effect of the premultiplier adds another $\pi/2$, which leads to a final phase of π. If the input is a logical "zero", the phase will decrease by $-\pi/2$. The effect of the premultiplier leads to a final phase state of zero. In other words, the premultiplier reduces the number of phase states from four ($+/- \pi/2$, 0, π) to two (0, π). The trellis phase diagram for MSK with premultiplication is illustrated in Figure 17.

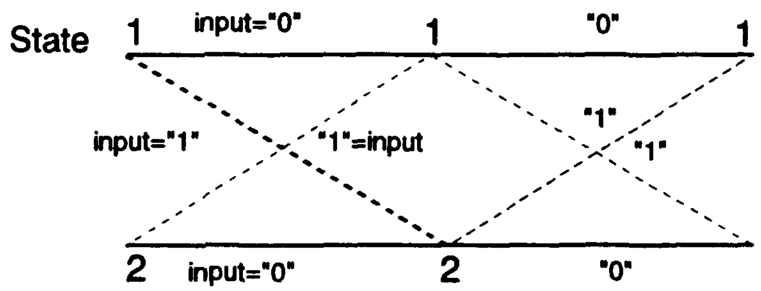

Figure 17. Phase Trellis Diagram for MSK After Premultiplication.

After the premultiplication, the received signal is correlated and each branch is used as an input to the Viterbi algorithm, as illustrated in the receiver block diagram shown in Figure 18. A soft Viterbi algorithm tracks the phase changes along the trellis and decides on the most probable path by considering as a decision rule the minimum euclidean distance to the four points in the two-dimensional (2-D) plane formed from the receiver output pairs. In this receiver four output pairs are possible, depending on the previous phase state and the input bit (see Figure 19).

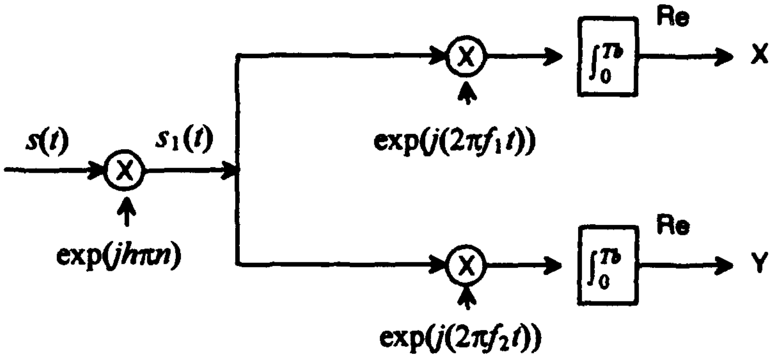

Figure 18. Receiver Block Diagram for Viterbi Decoding.

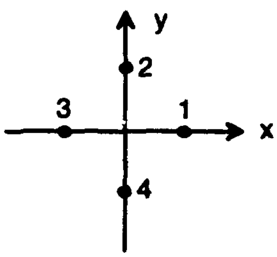

Figure 19. Possible Outputs from the Viterbi Demodulator.

Without noise, the output pair will coincide exactly with one of the four possible points, depending on the input bit and the previous phase state, as described in Table 13.

TABLE 13. POSSIBLE OUTPUT POINTS FROM THE VITERBI DEMODULATOR

	Previous State	
Input Bit	1	2
0	1	3
1	2	4

It is useful to notice that if the input is a "1", independent of the previous phase state, the output pair will be on the "y" axis, whereas if the input is a "0" the output will be on the "x" axis.

The simulation was run using MSK and the Viterbi receiver. The results were found to be a bit degraded (~0.5 dB) with respect to the receiver described in Figure 16, but still superior to DBPSK. The different codes that were used to simulate MSK can be found in Appendix B.

2. Coherent Reception of CPFSK with h=0.4

It was seen in the previous chapter that CPFSK with modulation index h=0.4 increases the C/I by a small amount and could therefore lead to better performance as far as this interference is concerned.

Since CPFSK with h=0.4 is not an orthogonal signaling set [Ref. 1], the first coherent receiver that was used to decode MSK cannot be used as a demodulator. However, the Viterbi algorithm can still determine a maximum likelihood path through the phase trellis diagram and optimally decode the signal. A Viterbi receiver was designed for the h=0.4 CPFSK signal. For this signal, there are five possible phase states

(0, +/- $2\pi/5$, +/- $4\pi/5$), and th ..nnot be reduced by premultiplication. The possible output pairs in the 2-D euclidean plane are therefore 10. The large number of points leads to a serious degradation of receiver performance since the points on the euclidean plane are very close to one another. When noise is added, a very high signal-to-noise ratio is required to avoid performance degradation. Since this is not the case for a satellite channel, CPFSK with h=0.4 cannot perform as well as MSK even though it has a very small advantage with respect to adjacent channel interference. The code that was written to simulate CPFSK (h=0.4) can be found in Appendix C.

3. Noncoherent Reception of MSK

Coherent reception is difficult to carry cut in terms of receiver complexity because carrier synchronization is required. It was decided to investigate the performance when noncoherent MSK is used. Several noncoherent receivers have been described in the literature [Ref. ., 9, 10, 11]. The best performance against noise is obtained by using the noncoherent receiver developed by Crozier, et. al. [Ref. 11]. A block diagram of the receiver can be seen in Figure 20.

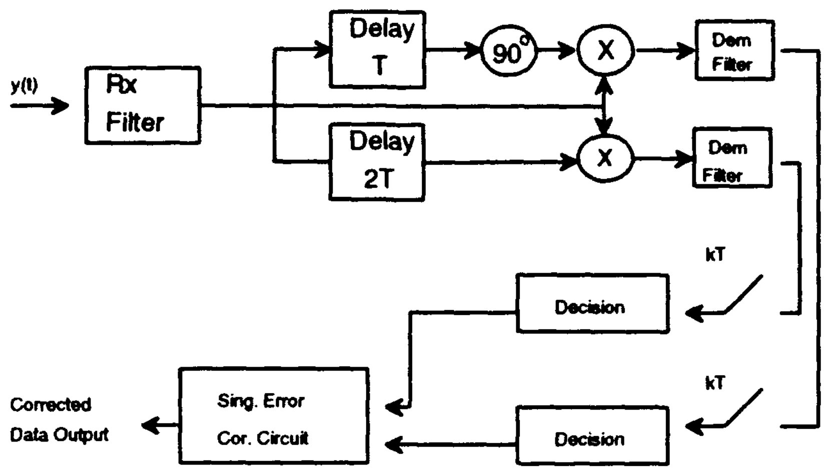

Figure 20. Noncoherent Receiver of MSK with Nonredundant Error Correction.

The receiver consists of a differential detection branch that measures the difference in phase between two successive signaling intervals, and a second branch where the symbol detected from the difference in phase between two alternate signaling intervals can be interpreted as the parity check sum of two successive transmitted data elements. These two symbols correspond to data and parity of a rate 1/2 single-error-correcting self-orthogonal convolutional code; therefore, performance can be improved by using the decoder for this error correcting code [Ref. 9].

To get even better performance, two filters are added. The reception filter is a 4-pole phase equalized Butterworth filter with filter-bandwidth-bit-duration product (BT)=1.1 and the demodulation filter is a 4-pole phase equalized Butterworth filter with BT=1.5.

The results of the simulation can be seen in Tables 13, 14, and 15.

TABLE 13. NUMBER OF ERRORS FOR THE SATELLITE CHANNEL WITH 14 dB Eb/No.

	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT CHANNELS (100 kHz)	0	0	0	1
ADJACENT CHANNELS (15 kHz)	0	0	0	84
ADJACENT CHANNELS (12.5 kHz)	1	2	5	157

TABLE 14. NUMBER OF ERRORS FOR THE SATELLITE CHANNEL WITH 12 dB Eb/No.

	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT CHANNELS (100 kHz)	0	0	0	1
ADJACENT CHANNELS (15 kHz)	1	0	1	90
ADJACENT CHANNELS (12.5 kHz)	3	10	20	174

TABLE 15. NUMBER OF ERRORS FOR THE SATELLITE CHANNEL WITH 10 dB Eb/No.

	2400 bps	4800 bps	9600 bps	19200 bps
ADJACENT CHANNELS (100 kHz)	0	0	0	1
ADJACENT CHANNELS (25 kHz)	0	0	0	2
ADJACENT CHANNELS (12.5 kHz)	3	10	20	174

From these results, it can be concluded that the noncoherent receiver would work in high signal-to-noise ratio situations. However, in a noisy channel the single error correction circuit cannot correct the data transmitted, and the receiver cannot perform even as well as DBPSK.

The code written to simulate noncoherent MSK can be found in Appendix D.

4. Coherent MSK Revisited

The coherent reception of MSK needs both a carrier recovery circuit and a clock recovery circuit. An example of a circuit suitable for this purpose can be found in the work of deBuda [Ref. 12]. The original circuit generates 90 degree phase and multiples of 90 degree phase ambiguity in the reference carrier phase. An improvement to the circuit that resolves the phase ambiguity of +/- 90 degrees can also be found in deBuda's work. One way to solve the remaining 180 degree phase ambiguity is by differentially encoding the bit stream. However, this last step is not necessary since the Viterbi algorithm resolves this ambiguity automatically. The trellis diagram remains the same when this ambiguity is introduced in the receiver, but the 0 and π phase states are interchanged, as shown in Figure 21.

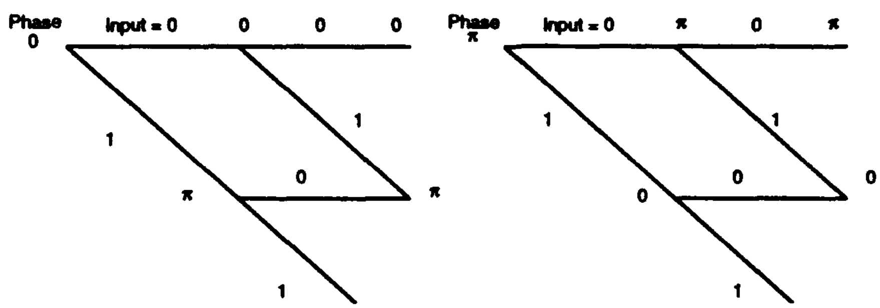

Figure 21. Comparative Trellis Diagram.

Assume the coherent references are shifted incorrectly by 180 degrees. The situation is pictured in Figure 22.

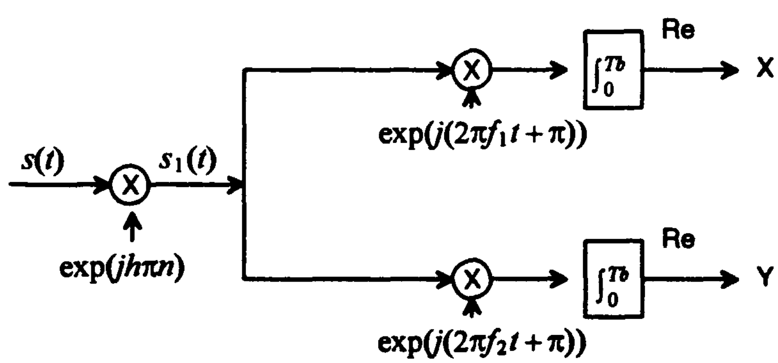

Figure 22. Viterbi Demodulator with Coherent References Shifted by 180 Degrees.

The output of each branch will be:

$$X = \int_0^{Tb} Re[s_1(t) * \exp(j * (2\pi f_1 t + \pi))]dt \; ; \qquad [10]$$

$$X = \int_0^{Tb} Re[(I(t) + jQ(t)) * (\cos(2\pi f_1 t + \pi) + j\sin(2\pi f_1 t + \pi))]dt \; ; \qquad [11]$$

$$X = \int_0^{Tb} (I(t)\cos(2\pi f_1 t + \pi) - Q(t)\sin(2\pi f_1 t + \pi))dt \; ; \qquad [12]$$

$$X = -[\int_0^{Tb} (I(t)\cos(2\pi f_1 t) - Q(t)\sin(2\pi f_1 t))dt] \; . \qquad [13]$$

In the same way,

$$Y = -[\int_0^{Tb} (I(t)\cos(2\pi f_2 t) - Q(t)\sin(2\pi f_2 t))dt] \; . \qquad [14]$$

But the terms in brackets are the outputs of the demodulator if the phases are not shifted. Therefore, since X and Y are the components in the 2-D plane of the output point, it is easy to see that the new output has been shifted by 180 degrees. From Table 13, if the output is shifted by 180 degrees, the Viterbi algorithm still decodes it as the same bit. Only 180 degree ambiguities can be resolved in this fashion.

In summary, the clock and carrier recovery circuit can be implemented as shown by deBuda [Ref. 12], however it is not necessary to differentially encode the message if a Viterbi algorithm is used as the decoder.

IV. CONCLUSIONS AND RECOMMENDATIONS

The results obtained in this thesis show that coherent minimum-shift keying with Viterbi decoding can improve the performance of a UHF satellite system when interference coming from adjacent channels is the main concern.

It was shown that continuous phase frequency-shift keying with modulation index other than h=0.5 and non-coherent reception of MSK are not suitable since in one way or another their performance is seriously degraded in a noisy environment.

For coherent MSK, a carrier recovery circuit that does not add great complexity to the receiver and that does not adversely affect the performance of the coherent MSK modulation is required. A circuit was presented that satisfies these criteria. It was demonstrated that the circuit's residual 180 degree phase ambiguity is solved by the Viterbi algorithm without differentially encoding the data.

Unfortunately, the results obtained in this work do not fully support the thesis that adjacent channel interference is limiting satellite channel bit rate since no interference-caused errors are observed in the simulation when the channels are separated by 100 kHz. The results are consistent with the limitations of the computer model and the high carrier-to-noise ratios computed for this system, and the work done for this thesis provides a good comparative idea of the behavior of the channel under those circumstances. However, it is necessary to have a more accurate tool to measure the actual performance of the satellite to determine whether coherent MSK has, in fact, any real benefits for UHF satellite communications.

APPENDIX A.

AWGN.M (AWGN FUNCTION)

```
        function y = awgn(x,sigma)
% Awgn is an M_file that adds awgn to the matrix x. The standard deviation of
% the noise is also an input (sigma) and it has to be change according to the
% different Eb/No that are desired to simulate, where Eb/No = 1/(2*sigma^2).
        [rr,cc] = size(x);
        seed = 0;
        rand('normal');
        rand('seed',seed);
        w = rand(rr,cc) + j*rand(rr,cc);
        y = x + sigma.*w;
```

COMPARE.M (NUMBER OF ERRORS FUNCTION)

```
        function out = compare(in,in1)
% This M_file accepts two vectors of equal length composed by zeros and ones
% and returns the number of bits in which both vectors do not agree.
        com = abs(in - in1);
        out = sum(com);
```

DBPSK.M (DBPSK MAIN PROGRAM)

```
% receiver for DBPSK
        m = 1002;
        rnd_o1 = msg(40,m); % Creating the random message
        rnd_o2 = msg(43,m); % Creating the interference sources.
        rnd_o3 = msg(65,m);
        dif_o1 = dif_cod(rnd_o1); % Differentially encoding the message.
        dif_o2 = dif_cod(rnd_o2); % Differentially encoding the interfering
                                  % messages
        dif_o3 = dif_cod(rnd_o3);
        map_o1 = map(dif_o1); % Mapping the message
```

```
map_o2 = map(dif_o2); % Mapping the interfering message
map_o3 = map(dif_o3);
dd = [38 38 38 41]; % Filter delays
T = [(1/2400) (1/4800) (1/9600) (1/19200)]; % Bit durations
t = 1/384000; % Sampling interval.
f1 = 0;
delta_f = 100000; % Frequency separation.
bit = [160 80 40 20];
sigma = [2*sqrt(2) 2 sqrt(2) 1]; % Standard deviation of the noise
clear dif_o1 dif_o2 dif_o3 md_o2 md_o3
for j=4:4,
  mod_sig1 = modul(map_o1,T(j),t,f1); % BPSK modulation
  mod_sig2 = modul(map_o2,T(j),t,delta_f);
  mod_sig3 = modul(map_o3,T(j),t,-delta_f);
  mod_sig = mod_sig1 + mod_sig2 + mod_sig3; % Adding the signals
   clear mod_sig1 mod_sig2 mod_sig3
  ch_sig = awgn(mod_sig,sigma(j)); % Adding the Gaussian noise
  clear mod_sig
  ch_sig = ch_sig';
  ch_sig = ch_sig(:);
  ch_sig = ch_sig';
  [b1,a1] = cheby1(6,.01,0.0651);
  prefil_sig = filter(b1,a1,ch_sig); % Prefiltering the signal
  clear ch_sig
  lim_sig = limiter(prefil_sig); % Hard limiter effect
  clear prefil_sig
  [b2,a2] = cheby1(4,.025,0.0394);
  postfilsig = filter(b2,a2,lim_sig); % Postfiltering the signal
  clear lim_sig
  num = length(postfilsig);
  postfilsig=[postfilsig(1,dd(j):num) postfilsig(1,1:(dd(j)-1))];% Filter
  sig_in = reshape(postfilsig,bit(j),m+1);                % delay
  clear postfilsig
  sig_in = conj(sig_in');
  rec_sig = demod(sig_in,m); % BPSK demodulation
  clear sig_in
  errors(j) = compare(md_o1(1:m-2),rec_sig(1:m-2)); % Checking errors
  clear rec_sig
end
diary juan.d
errors  % Saving the results in a diary file
```

diary off

DBPSPO.M (DBPSK ADJACENT CHANNEL INTERFERENCE POWER COMPUTATION)

% This M_file computes the power in the main channel and the power of the
% adjacent channel that is leaking into the main channel. After that a ratio
% between both powers is obtained.

```
m = 1000; % Number of bits
md_o = msg(10,m); % Random message generation
md_o1 = msg(25,m);
dif_o = dif_cod(md_o); % Differentially encoding the message
dif_o1 = dif_cod(md_o1);
map_o = map(dif_o); % Mapping the message
map_o1 = map(dif_o1);
delta_f = 100000; % Separation between channels
f1 = 0;
clear md_o dif_o
t = 1/384000; % Sampling interval
T = [(1/2400) (1/4800) (1/9600) (1/19200)]; % Bit durations
for j=1.4,
  mod_sig = modul(map_o,T(j),t,delta_f); % BPSK modulation
  mod_sig1 = modul(map_o1, T(j),t,f1);
  mod_sig = mod_sig';
  mod_sig = mod_sig(:);
  mod_sig = mod_sig';
  mod_sig1 = mod_sig1';
  mod_sig1 = mod_sig1(:);
  mod_sig1 = mod_sig1';
  [b1,a1] = cheby1(6,.01,0.0651);
  prefil_sig = filter(b1,a1,mod_sig);  % Prefiltering the signal

  prefil_sig1 = filter(b1,a1,mod_sig1);
  lim_sig = limiter(prefil_sig); % Hard limiting the signal
  lim_sig1 = limiter(prefil_sig1);
  clear prefil_sig prefil_sig1
  [b2,a2] = cheby1(4,.025,0.0394);
  postfil_sig = filter(b2,a2,lim_sig); Postfiltering the signal
  postfil_sig1 = filter(b2,a2,lim_sig1);
  clear lim_sig lim_sig1
  ll = length(postfil_sig);
  power_ad(j) = sum(abs(postfil_sig).^2)/ll; % Computing the power
  power_base(j) = sum(abs(postfil_sig1).^2)/ll;
```

```
  clear postfil_sig postfil_sig1
  clear mod_sig mod_sig1
  norm_power(j) = power_base(j)/power_ad(j); % Computing the ratio
  norm_pow_dB(j) = 10*log10(norm_power(j));
end
diary juan.d
norm_pow_dB % Saving the results in a diary file
diary off
```

DEMOD.M (DBPSK DEMODULATION FUNCTION)

```
function out = demod(in,m)
% This M_file performs noncoherent demodulation of DBPSK. The matrix in
% contains the sampled DBPSK waveform and m is the number of bits that this
% waveform represents.
o = ones(1,m);
for i=2:m+1,
  dif(i-1) = abs(sum(in(i,:)) - sum(in(i-1,:)));
  su(i-1) = abs(sum(in(i,:)) + sum(in(i-1,:)));
  metric(i-1) = dif(i-1) - su(i-1);
  if metric(i-1) < 0,
    o(i-1) = 0;
  end
end
out = o;
```

DIF_COD.M (DIFFERENTIALLY ENCODING FUNCTION)

```
function dif_o = dif_cod(in)
% This M_File differentially encodes a bit stream that is input in the
% variable in.
a = length(in);
y = [1, zeros(1,a)];
for i=1:a,
  y(i+1) = xor(in(i),y(i));
end
dif_o = y;
```

ESTBPS.M (ESTIMATION OF CARRIER TO INTERFERENCE RATIO)

% This is the main program to estimate the carrier to interference ratio. In
% this case the modulation used is DBPSK but the method holds for any
% modulation scheme.

```
m = 1000; % Number of bits
md_o = msg(40,m); % creating the random message.
md_o1 = msg(65,m); % creating the interfering message.
dif_o = dif_cod(md_o); %differentially encoding the message
dif_o1 = dif_cod(md_o1);
map_o = map(dif_o); % mapping the message.
map_o1 = map(dif_o1);
T = [(1/2400) (1/4800) (1/9600) (1/19200)]; % Bit durations
t = 1/384000; % Sampling interval.
delta_f = 100000; % frequency separation
f1 = 0;
clear md_o dif_o md_o1 dif_o1
for j=1:4,
  mod_sig = modul(map_o,T(j),t,delta_f); % DBPSK modulation
  mod_sig1 = modul(map_o1,T(j),t,f1);
  inpass_b = mod_sig + mod_sig1; % Adding both messages
  inpass_b = inpass_b';
  inpass_b = inpass_b(:);
  inpass_b = inpass_b';
  mod_sig1 = mod_sig1';
  mod_sig1 = mod_sig1(:);
  mod_sig1 = mod_sig1';
  [b1,a1] = cheby1(6,.01,0.0651);
  prefil_sig = filter(b1,a1,inpass_b); % Filtering both messages
   prefil_sig1 = filter(b1,a1,mod_sig1); % Filtering the main message
  lim_sig = limiter(prefil_sig); % Hard limiting both messages
  clear prefil_sig
  [b2,a2] = cheby1(4,.025,0.0394);
  postfil_sig = filter(b2,a2,lim_sig);
  postfil_sig1 = filter(b2,a2,prefil_sig1);
  clear lim_sig prefil_sig1
  dd = sum(postfil_sig1.^2);
  gain(j) = sum(postfil_sig.*postfil_sig1)/dd; % computing the GAIN
```

```
  in_estimate = gain(j)*postfil_sig1; % computing the baseband estimate
  band_estimate = postfil_sig - in_estimate; % computing the interfering
  % estimate
  clear postfil_sig postfil_sig1
  ff = length(in_estimate);
  power_in(j) = (sum(abs(in_estimate).^2))/ff
  power_band(j) = (sum(abs(band_estimate).^2))/ff
  C_to_I(j) =10*log10(power_in(j)/power_band(j)); % computing the C/I
  clear in_estimate band_estimate
  clear mod_sig mod_sig1 inpass_b
end
diary juan.d
C_to_I % Saving the results in a diary file.
diary off
```

LIMITER.M (HARD LIMITER FUNCTION)

```
function out=limiter(in)
```

% This M_File performs a hard limiting effect over a modulated signal. This
% signal is contained in the vector in

```
ss = abs(in);
out = in./ss;
```

MAP.M (MAPPING FUNCTION)

```
function out = map(in)
```

% This M_File maps a bit stream to 0 or pi to be able to perform afterwards
% a BPSK modulation.

```
a = length(in);
for i=1:a,
 if (in(i) == 0),
  out(i) = 0;
 else
  out(i) = pi;
 end
end
```

MODUL.M (BINARY PHASE SHIFT KEYING MODULATION FUNCTION)

```
function out = modul(in,T,t,fc)
% This M_File performs BPSK modulation.It accepts the signal in, the bit
% duration T, the sampling interval t and the carrier frequency fc as inputs
time = 0:t:(T-t);
a = length(in);
p = 1;
for s=1:a,
  time = time + (p - 1)*T;
  p = 2;
  if in(s) == pi,
    out(s,:) = exp(j*(2*pi*fc*time + pi));
  else
    out(s,:) = exp(j*(2*pi*fc*time));
  end
end
```

MSG.M (MESSAGE GENERATION FUNCTION)

```
function u = msg(seed,k)
% This M-file accepts a data vector with seed for rand and
%k the number of bits that will be returned in the vector u
rand('uniform')
rand('seed',seed)
u = round(rand(1,k));
```

XOR.M (EXCLUSIVE-OR FUNCTION)

```
function a = xor(in,in1)
% This M_File performs the xor logic operation
if in == in1,
  a = 0;
else
  a = 1;
end
```

APPENDIX B.

CODEMOD.M (COHERENT MSK DEMODULATION FUNCTION)

```
        function out = codemod(in,t,T,fc,h,m)
% This function performs coherent demodulation of Minimum Shift Keying using
% correlation, sampling and integration in each of the two branches of the
% receiver. The integration is performed over a period equal to twice the bit
% duration and the decision is made by alternatively evaluate the output of
% the two branches.
        time = 0:t:(T-t);
        dd = 1;
        ff = 1;
        for s=1:m,
          phi1 = cos(pi*time*h/T);
          phi2 = sin(pi*time*h/T);
          if rem(s,2) ~= 0,
            vec1(dd,:) = in(s,:).*phi1;
            vec2(dd,:) = in(s,:).*phi2;
            dd = dd + 1;
          else
            vec3(ff,:) = in(s,:).*phi1;
            vec4(ff,:) = in(s,:).*phi2;
            ff = ff + 1;
          end
          time = time + T;
        end
        vec2 = vec2 + vec4;
        vec2 = vec2';
        sec = sum(vec2);
        last = sum(vec3(ff-1,:));
        [rr cc] = size(vec1);
        vec1 = vec1(2:rr,:);
        vec3 = vec3(1:rr-1,:);
        vec1 = vec1 + vec3;
        vec1 = vec1';
        one = sum(vec1);
        one = [one last];
```

```
for ee=1:m/2,
   if real(one(ee)) > 0,
      est1(ee) = 0;
   else
      est1(ee) = pi;
   end
   if imag(sec(ee)) > 0,
      est2(ee) = -pi/2;
   else
      est2(ee) = pi/2;
   end
end
if est2(1) == -pi/2,
  dec(1) = 0;
else
  dec(1) = 1;
end
k = 1;
v = 1;
for gg=2:m,
  if rem(gg-1,2) ~= 0,
    if (est1(k)==0 & est2(k)==-pi/2) | (est1(k)==pi & est2(k)==pi/2),
      dec(gg) = 0;
    else
      dec(gg) = 1;
    end
    k = k + 1;
  end
  if rem(gg-1,2) == 0,
    if (est1(v)==0 & est2(v+1)==-pi/2) | (est1(v)==pi & est2(v+1)==pi/2),
      dec(gg) = 0;
    else
      dec(gg) = 1;
    end
    v = v + 1;
  end
end
out = dec;
```

CPFSKMOD.M (CONTINUOUS PHASE FREQUENCY SHIFT KEYING MODULATION FUNCTION)

```
function out = cpfskmod(in,T,t,fc,h)
% This M_file performs the modulation of CPFSK with any modulation index
% since it accepts h as an input.
teta0 = 0;
a = length(in);
time = 0:t:(T-t);
for s=1:a,
  if s == 1,
    teta = 0;
  else
    teta0 = teta0 + in(s-1);
    teta = pi*h*teta0;
  end
  time = time + T;
  if in(s) == -1,
    f1 = fc - (h/(2*T));
    mod_output(s,:) = exp(j*(2*pi*f1*time + teta + s*pi*h));
  else
    f2 = fc + (h/(2*T));
    mod_output(s,:) = exp(j*(2*pi*f2*time + teta - s*pi*h));
  end
end
out = mod_output';
out = out(:);
out = out';
```

EUCDIS.M (EUCLIDEAN DISTANCE FUNCTION)

```
function D = eucdis(q,R)

% This M-file finds Euclidean distance of elements in vector R from
% q unit amplitude vectors equally spaced on the unit circle. It stores
% these as rows of D.
L = length(R);
index = 1:q;
dph = 2*pi/q;
MO = exp(j*(dph.*(index-1)));
for l=1:L,
```

```
        D(l,:) = abs(R(l).*ones(MO) - MO);
      end
```

MAPPER.M (MAPPING FUNCTION)

```
      function output = mapper(in);
% This M_File maps the bit stream to 1's or -1's to be able to perform MSK
% modulation afterwards.
      k = in == 0;
      k = -k;
      output = k + in;
```

MATCH.M (OFFSET FUNCTION)

```
      function [out,out1] = match(N,in,in1)
% This M_File matches vectors in and in1 which are offset by N positions.
      if length(in) == length(in1),
        out = in(1:length(in) - N);
        out1 = in1(N+1:length(in1));
      end
```

MSKVI.M (MSK RECEIVER WITH VITERBI ALGORITHM)

```
% receiver for MSK with Viterbi decoding
      clear
      m = 1020;
      diary juan.d
      md_o1 = msg(40,m); % Creating the mesage
      map_o1 = mapper(md_o1); % Mapping the function
      T = [(1/2400) (1/4800) (1/9600) (1/19200)]; % Bit duration
      t = 1/384000; % Sampling interval
      f1 = 0;
      h = 0.5; % Modulation index
      sigma = [2*sqrt(2) 2 sqrt(2) 1]; % Standard deviation of the noise
      clear md_o2 md_o3
```

```
for kk=1:4,
  mod_sig1 = cpfskmod(map_o1,T(kk),t,f1,h); % MSK modulation
  ch_sig = awgn(mod_sig1,sigma(kk)); % Adding the Gaussian noise

  clear mod_sig1
  dem_sig = videmod1(ch_sig,t,T(kk),h); % Mapping the signal to the
                                        % euclidean plane
  clear ch_sig
  TT = zeros(2,60); % Input matrix to the Viterbi Algorithm
  vipath = [1 0 1 2 1 4; 1 1 2 2 0 3];
  for qq=1:m,
     D = dem_sig(qq,:);
     TT = softv(1,2,20,TT,vipath,D); % Viterbi algorithm function
     vi_sig(qq) = TT(1,60);
  end
  clear dem_sig
  [mes_o1,rec_sig] = match(19,md_o1,vi_sig); % Offset function
  errors(kk) = compare(mes_o1,rec_sig); % Checking errors
  clear rec_sig mes_o1 vi_sig
end
errors % Saving the results in a diary file
diary off
```

REMSK.M (COHERENT MSK MAIN PROGRAM)

```
% receiver for MSK
      m = 1000;
      diary juan.d
      md_o1 = msg(40,m); % Creating the message
      md_o2 = msg(43,m); % Creating the interfering message
      md_o3 = msg(65,m);
      map_o1 = mapper(md_o1); % Mapping the message
      map_o2 = mapper(md_o2); % Mapping the interfering message
      map_o3 = mapper(md_o3);
      T = [(1/2400) (1/4800) (1/9600) (1/19200)]; % Bit durations
      t = 1/384000; % Sampling interval
      f1 = 0;
      delta_f = 100000; % Frequency separation
      h = 0.5; % Modulation index
```

```
sigma = [2*sqrt(2) 2 sqrt(2) 1]; % Standard deviation of the noise
bit = [160 80 40 20];
dd = [38 38 38 41]; % Filter delay
clear md_o2 md_o3
for kk=1:4,
  mod_sig1 = cpfskmod(map_o1,T(kk),t,f1,h); % MSK message
modulation
  mod_sig2 = cpfskmod(map_o2,T(kk),t,delta_f,h); % MSK interfering
message
  mod_sig3 = cpfskmod(map_o3,T(kk),t,-delta_f,h); % modulation
  mod_sig = mod_sig1 + mod_sig2 + mod_sig3; % Adding the
messages
  clear mod_sig1 mod_sig2 mod_sig3
  ch_sig = awgn(mod_sig,sigma(kk)); % Adding the Gaussian noise

  clear mod_sig
  ch_sig = ch_sig';
  ch_sig = ch_sig(:);
  ch_sig = ch_sig';
  [b1,a1] = cheby1(6,.01,0.0651);
  prefil_sig = filter(b1,a1,ch_sig); % Prefiltering the signal
  clear ch_sig
  lim_sig = limiter(prefil_sig); % Hard limiter effect
  clear prefil_sig
  [b2,a2] = cheby1(4,.025,0.0394);
  postfilsig = filter(b2,a2,lim_sig); % Postfiltering the signal
  clear lim_sig
  num = length(postfilsig);
  postfilsig = [postfilsig(1,dd(j):num) postfilsig(1,1:dd(j)-1)]; % Filter
  sig_in = reshape(postfilsig,bit(kk),m);                        % delay
  clear postfilsig
  sig_in = conj(sig_in');
  rec_sig = codemod(sig_in,t,T(kk),f1,h,m); % Coherent demodulation of
                                            % MSK
  clear sig_in
  errors(kk) = compare(md_o1,rec_sig); % Checking errors
  clear rec_sig
end
errors % Saving the result in a diary file
diary off
```

SOFTV.M (SOFT VITERBI DECODER)

```
          function PHN = softv(k,K,Np,PH,T,D)
%         Soft Viterbi Decoder
%           Paul H. Moose
%       Univ. degli Studi di Padova
%             17-05-91
% This M-file decodes k bit msgwords from 2^n real metrics
% (These may, for example, represent the "distance" of the
% received modulation value from each of 2^n modulation
% values.)
% The state transition information for a 2^K state trellis is in
% the 2^K by 3*2^k matrix T. Each of the 2^k entering paths to
% each state has its source state (one of 2^K), path msgwords (one
% of 2^k) and path codeword (one of 2^n) listed in the state row.
% The path histories are kept in matrix PH that is 2^K by 3*Np.
% The path history for each state contains source state, path
% weight and path codeword for Np previous states.
% The output PHN is the update of PH, the new path history.
% The decoded codeword is in the last column of PHN. (They should
% "merge").
% The past histories are updated on the basis of the "minimum
% metric". You can change this to the "maximum metric" if desired as
% indicated in  the comments in the code.
%
          for ff=1:K
            X(ff,2) = D(T(ff,3)) + PH(T(ff,1),2); %path weight
            X(ff,1) = T(ff,1); %path source state
            X(ff,3) = T(ff,2); %path code word T(ff,3).Chg to T(ff,2) for msgword
              for l=2:2^k
                wt = D(T(ff,3*l)) + PH(T(ff,3*l-2),2);
                if wt < X(ff,2) % The < selects min metric
                  X(ff,2) = wt;
                  X(ff,1) = T(ff,3*l-2);
                  X(ff,3) = T(ff,3*l-1); % Ghg to T(ff,3*l) for codeword
                end
              end
% We need now to append old paths to new paths to get survivors.
            PHN(ff,:) = [X(ff,:) PH(X(ff,1),1:3*Np-3)];
          end
```

VIDEMOD1.M (VITERBI DEMODULATION FUNCTION)

```
        function out = videmod1(in,t,T,h)
% This M_File accepts a modulated signal and matches it on the euclidean
% plane. The euclidean distance from these points to 4 different points
% is found and the metric is returned to be used as an input in the soft
% Viterbi decoder
        no_mat = [];
        map = [-1 1];
        [rr cc] = size(in);
        time = 0:t:T-t;
        for ss=1:rr,
          in(ss,:) = in(ss,:)*exp(j*(ss-1)*pi*h);
          for m_ary=1:2,
            xx = exp(j*pi*h*time*map(m_ary)/T);
            first = xx*conj(in(ss,:)');
            no_mat = [no_mat first];
          end
          demod(ss,:) = real(no_mat);
          no_mat = [];
        end
        R = j*demod(:,1) + demod(:,2);
        out = eucdis(4,R);
```

APPENDIX C.

CPFSDIS.M (EUCLIDEAN DISTANCE FUNCTION FOR CPFSK WITH h = 0.4)

```
        function D = cpfsdis(R)
% This M-file finds Euclidean distance of elements in vector R from
% 10 unit amplitude vectors on the unit circle (This is the case of CPFSK
% with h=0.4). It stores these as rows of D.
        L = length(R);
        dph = [0.235 1.336 1.183 2.672 2.867 -2.221 -2.491 -1.296 -1.101
        0.388];
        MO = exp(j*dph);
        for l=1:L,
           D(l,:) = abs(R(l).*ones(MO) - MO);
        end
```

CPFSK.M (VITERBI ALGORITHM RECEPTION OF CPFSK)

```
% receiver for CPFSK with h= 0.4
        m = 1000;
        diary juan.d
        md_o1 = msg(40,m); % Creating the message
        map_o1 = mapper(md_o1); % Mapping the message
        T = [(1/2400) (1/4800) (1/9600) (1/19200)]; % Bit durations
        t = 1/384000; % Sampling interval
        f1 = 0;
        h = 0.4; % Modulation index
        sigma = [2*sqrt(2) 2 sqrt(2) 1]; % Standard deviation of the noise
        clear md_o2 md_o3
        for kk=1:4,
          mod_sig1 = cpfskmod(map_o1,T(kk),t,f1,h); % CPFSK modulation
          ch_sig = awgn(mod_sig1,sigma(kk)); % Adding the Gaussian noise

          clear mod_sig1
          dem_sig = videmod(ch_sig,t,T(kk),h); % Mapping the signal in the
                                                % euclidean plane
```

```
  clear ch_sig
  TT = zeros(5,60); % Input matrix to the Viterbi algorithm
  vipath=[2 0 3 5 1 10; 3 0 5 1 1 2; 4 0 7 2 1 4; 5 0 9 3 1 6; 1 0 1 4 1 8];
  for qq=1:m,
    D = dem_sig(qq,:);
    TT = softv(1,5,20,TT,vipath,D); % Viterbi decoding function
    vi_sig(qq) = TT(1,60);
  end
  clear dem_sig
  [mes_o1,rec_sig] = match(19,md_o1,vi_sig); % Offset function
  errors(kk) = compare(mes_o1,rec_sig); % Checking errors
  clear rec_sig mes_o1 vi_sig
end
errors % Saving the results in a diary file
diary off
```

VIDEMOD.M (VITERBI DEMODULATION FUNCTION)

```
function out = videmod(in,t,T,h)
% This M_File accepts a modulated signal and matches it on the euclidean
% plane. The euclidean distance from these points to 10 different points
% is found and the metric is returned to be used as an input in the soft
% Viterbi decoder
no_mat = [];
map = [-1 1];
[rr cc] = size(in);
time = 0:t:T-t;
for ss=1:rr,
  for m_ary=1:2,
    xx = exp(j*pi*h*time*map(m_ary)/T);
    first = xx*conj(in(ss,:)');
    no_mat = [no_mat first];
  end
  demod(ss,:) = real(no_mat);
  no_mat = [];
end
R = j*demod(:,1) + demod(:,2);
out = cpfsdis(R);
```

APPENDIX D.

AND.M (AND GATE FUNCTION)

```
function out = and(in,in1)
% This M_file accepts two bits as inputs and performs the logical "and" operation
% between them.
if in == 1 & in1 == 1,
  out = 1;
else
  out = 0;
end
```

DECISION.M (DECISION BLOCK FOR NONCOHERENT MINIMUM SHIFT KEYING)

```
function out=decision(in)
% This M_file accepts a vector that represents the output of the integrator
% in noncoherent reception of MSK and decides whether this output corresponds
% to a zero or a one.
a = length(in);
out = zeros(1,a);
for j=1:a,
   if in(j) > 0,
     out(j) = 1;
   end
end
```

MSKDEMOD.M (NONCOHERENT MSK DEMODULATION FUNCTION)

```
function out = mskdemod(in)
% This M_File performs noncoherent MSK demodulation over a signal contained
% in the matrix in
for h=2:length(in),
```

```
    y(h,:) = real(in(h,:).*conj(in(h-1,:)).*exp(-j*(pi/2)));
    end
  out = y;
```

PARITY.M (PARITY BIT FUNCTION)

```
function out1 = parity(in)
```

% This M_File obtains a parity bit from a MSK signal. This parity bit is
% going to be used in the single error correction circuit.

```
ss = length(in);
for h=3:ss,
  y1(h,:) = real(in(h,:).* conj(in((h-2),:)));
end
out1 = y1;
```

RENCMSK.M (NONCOHERENT RECEPTION OF MSK)

% receiver for NON COHERENT MSK

```
diary juan.d
m = 1005;
md_o1 = msg(40,m); % Creating the main message
md_o2 = msg(43,m); % Creating the interfering messages
md_o3 = msg(65,m);
map_o1 = mapper(md_o1); % Mapping the message
map_o2 = mapper(md_o2); % Mapping the interfering messages
map_o3 = mapper(md_o3);
dd = [71 55 45 43]; % Filter delays
T = [(1/2400) (1/4800) (1/9600) (1/19200)]; % Bit durations
t = 1/384000; % Sampling interval
f1 = 0;
delta_f = 100000; % Frequency separation
bit = [160 80 40 20];
sigma = [2*sqrt(2) 2 sqrt(2) 1]; % Standard deviation of the noise
h = 0.5; % Modulation index
B = [0.0138 0.0275 0.055 0.11]; % Bandwidth of the first Butterworth
filter
B1 = [0.0188 0.0375 0.075 0.15]; % Bandwidth of the sec. Butterworth
```

```
                    % filter
clear md_o2 md_o3
for j=1:4,
  mod_sig1 = cpfskmod(map_o1,T(j),t,f1,h); % MSK modulation
  mod_sig2 = cpfskmod(map_o2,T(j),t,delta_f,h);
  mod_sig3 = cpfskmod(map_o3,T(j),t,-delta_f,h);
  mod_sig = mod_sig1 + mod_sig2 + mod_sig3; % Adding the signals

  clear mod_sig1 mod_sig2 mod_sig3
  ch_sig = awgn(mod_sig,sigma(j)); % Adding the Gaussian noise
  clear mod_sig
  ch_sig = ch_sig';
  ch_sig = ch_sig(:);
  ch_sig = ch_sig';
  [b1,a1] = cheby1(6,.01,0.0651);
  prefil_sig = filter(b1,a1,ch_sig); % Prefiltering the signal
  clear ch_sig
  lim_sig = limiter(prefil_sig); % Hard limiter effect
  clear prefil_sig
  [b2,a2] = cheby1(4,0.025,0.0394);
  postfilsig = filter(b2,a2,lim_sig); % Postfiltering the signal
  clear lim_sig
  [b3,a3] = butter(4,B(j));
  recfil = filter(b3,a3,postfilsig); % Butterworth filter in the receiver
  clear b3 a3  postfilsig
  shapesig = reshape(recfil,bit(j),m);
  shapesig = conj(shapesig');
  demsig = mskdemod(shapesig); % MSK demodulation
  parsig = parity(shapesig); % Parity bit creation
  clear recfil shapesig
  demsig = demsig';
  demsig = demsig(:);
  demsig = demsig';
  parsig = parsig';
  parsig = parsig(:);
  parsig = parsig';
  [b4,a4] = butter(4,B1(j));
  demsig = filter(b4,a4,demsig); % Second Butterworth filter
  parsig = filter(b4,a4,parsig);
  demsig = [demsig(1,dd(j):length(demsig)) demsig(1,1:dd(j)-1)]; % Filter
  parsig = [parsig(1,dd(j):length(parsig)) parsig(1,1:dd(j)-1)]; % delay
```

```
    demsig = reshape(demsig,bit(j),m);
    parsig = reshape(parsig,bit(j),m);
    shapedem = sum(demsig);
    shapedem1 = shapedem(3:m);
    shapepar = sum(parsig);
    shapepar1 = shapepar(3:m);
    clear demsig
    dataout = decision(shapedem1); % Decision block
    paraout = decision(shapepar1);
    % Single error correction circuit
    datar = sinerror(dataout(2:length(dataout)),paraout(2:length(paraout)));
    clear dataout paraout
    clear shapedem shapedem1 shapepar shapepar1
    errors(j) = compare(rnd_o1(4:m-2),datar(2:length(datar))); %Checking
                                                                  % errors
    clear datar
  end
  errors
  % Saving the results in a diary file
  diary off
```

SINERROR.M (SINGLE ERROR CORRECTION FUNCTION)

```
  function correct = sinerror(in,in1)
% This M_File performs a single error correction accepting as inputs a vector
% in which contains the data and a vector in1 which contains the parity bits
  in = [0 0 0 in];
  in1 = [0 0 0 in1];
  out = zeros(1,length(in));
  out3 = zeros(1,length(in));
  out2 = zeros(1,length(in));
  for kk=3:length(in),
     out(kk) = xor(in(kk),in(kk-1));
     out1(kk) = xor(out(kk),in1(kk));
     out2(kk) = xor(out1(kk),out3(kk-1));
     out3(kk) = and(out2(kk-1),out1(kk));
     correct(kk-1) = xor(out3(kk),in(kk-1));
  end
  correct = correct(4:length(in)-1);
```

LIST OF REFERENCES

1. Couch, Leon III, *Digital and Analog Communications Systems*, McMillan, 1990.

2. Ha, Tri T., *Digital Satellite Communications*, McGraw Hill, 1990.

3. Hughes Aircraft Company Space and Communications Group, "Proposal for UHF Follow-On Communications Satellite, Vol. II," January, 1988.

4. Murota, K., and Hirade, K., "GMSK Modulation for Digital Mobile Radio Telephony," *IEEE Trans. Commun.*, Com-29, No. 7, July, 1981.

5. Kahnaman, Betulham, "Performance Evaluation of UHF Fading Satellite Channel by Simulation for Different Modulation Schemes," Master's Thesis, Naval Postgraduate School, Monterey, CA, December, 1992.

6. Haykin, S., *Communications Systems*, 2nd ed., Wiley and Sons, 1983.

7. Proakis, J.G., *Digital Communications*, 2nd ed., McGraw Hill, 1989.

8. Mason, Lloyd J., "Analysis and Simulation of a DMSK Receiver," *SCC 1983 Conf. Rec.*, Ottawa, June, 1982.

9. Masamura, T., Samejima, S., Morihiro, Y., and Fuketa, H., "Differential Detection of MSK with Nonredundant Error Correction," *IEEE Transactions on Communications*, Vol. Com-27, No. 6, June, 1979.

10. Bhargava, Haccoun, Matyas, and Nuspl, *Digital Communications by Satellite*, 1st ed., John Wiley and Sons, 1981.

11. Crozier, S., Mazur, B., and Matyas, R., "Performance Evaluation of Differential Detection of MSK," *GLOBECOM 82 IEEE Telecommun. Conf. Rec.*, Miami, Nov.-Dec., 1982.

12. deBuda, Rudi, "Coherent Demodulation of Frequency-Shift-Keying with Low Deviation Ratio," *IEEE Transactions on Communications*, June, 1972.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5000	2
3.	Director Space and Electronic Combat Division (N64) Space and Electronic Warfare Directorate Chief of Naval Operations Washington, DC 93943-5000	1
4.	Chairman, Electronic Warfare Academic Group, Code EW Naval Postgraduate School Monterey, CA 93943-9528	1
5.	Professor P.H. Moose, Code EC/Me Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Professor R. Clark Robertson, Code EC/Rc Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Servicio De Guerra Electrónica Comando De Operaciones Navales Puerto Belgrano, CP 8000, Argentina	1
8.	Sevicio De Inteligencia Naval Estado Mayor General De La Armada Buenos Aires, Argentina	1

9. Dirección De Instrucción Naval 1
Estado Mayor General De La Armada
Buenos Aires, Argentina